2015 IEEE 35th International Conference on Distributed Computing Systems

An Efficient Anonymous Authentication Protocol for RFID Systems Using Dynamic Tokens

Min Chen Shigang Chen
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611, USA

***Abstract*—Radio frequency identification (RFID) technologies are widely used in many applications. The widespread use of tags in traditional ways of deployment raises a privacy concern: They make their carriers trackable. This paper studies the problem of anonymous authentication. Due to resource constraints of low-cost tags, we develop a new technique to generate dynamic tokens for anonymous authentication by following an asymmetric design principle that pushes most complexity to more powerful RFID readers. Instead of implementing complicated cryptographic hash functions, our authentication protocol only requires tags to perform several simple hardware-efficient operations such as bitwise XOR, one-bit left circular shift and bit flip. Moreover, our protocol reduces the communication overhead and online computation overhead to $O(1)$ per authentication for both tags and readers, which compares favorably with the prior art.**

I. Introduction

To protect the privacy of tag carriers, we need to invent ways of keeping the usefulness of tags while doing so anonymously. One important problem is *anonymous authentication*, which is to authenticate a tag without requiring the tag to transmit any identifying information, such as tag ID, key identifier or any fixed number that may be used for identification purpose. In this paper, we make a fundamental shift from the traditional paradigm for anonymous authentication protocol design. On the one hand, we want to avoid implementing any complicated functions such as cryptographic hash on RFID tags due to hardware constraint. On the other hand, given the significant capability disparity between readers and tags, we follow an asymmetry design principle when designing our protocol: We should push most workload to the readers while leaving the tags as simple as possible. Based on this principle, we develop a new technique to generate dynamic tokens for anonymous authentication in RFID systems. Our protocol only requires the tags to perform a few hardware-efficient operations such as bitwise XOR, one-bit left circular shift and bit flip to randomize authentication data, while all other work is done by the readers. Moreover, our protocol reduces the communication overhead and online computation overhead to $O(1)$ per authentication for both readers and tags.

II. Dynamic Token based Authentication Protocol

Our Dynamic Token based Authentication Protocol (DTAP) consists of three phases as follows.

A. Initialization Phase

The central server stores all tags' keys in a *key table*, denoted by KT. As shown in Table I, each entry in KT is indexed by the *tag index*, supporting random access in $O(1)$ time. When t first joins the system, the reader randomly generates an array of a-bit base tokens $[bt^1, bt^2, \ldots, bt^m]$, and an array of b-bit base indicators $[bi^1, bi^2, \ldots, bi^n]$ for t. In addition, the a-bit token tk and b-bit indicator ic of t are also randomly initialized. After that, the reader requires the central server to insert those keys to KT. In addition, the central server maintains a hash table, denoted by HT, that maps the token of each tag to its tag index. At first, every slot in HT is initialized to zero. After t joins the system, the reader computes $h(tk)$ (h() is a hash fucntion), and then inserts the tag index idx of t into the $h(tk)$th slot of the hash table, i.e., $HT[h(tk)] = idx$. Fig. 2 shows the hash table built from the five tokens of tags t_1, t_2, t_3, t_4 and t_5.

Tag Index	Tag	Base Token Array	Token	Base Indicator Array	Indicator
1	t_1	$[bt_1^1, \ldots, bt_1^m]$	tk_1	$[bi_1^1, \ldots, bi_1^n]$	ic_1
2	t_2	$[bt_2^1, \ldots, bt_2^m]$	tk_2	$[bi_2^1, \ldots, bi_2^n]$	ic_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	t_N	$[bt_N^1, \ldots, bt_N^m]$	tk_N	$[bi_N^1, \ldots, bi_N^n]$	ic_N

TABLE I: Key table stored by the central server.

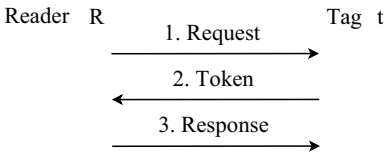

Fig. 1: Three steps of token-based mutual authentication.

B. Authentication Phase

The three-step authentication phase of DTAP is shown in Fig. 1. The reader R initiates the authentication process by sending a request, and the tag t responds with its token. After receiving the token tk, R first calculates $h(tk)$, and then accesses $HT[h(tk)]$ to retrieve the tag index, which is idx. If R finds $idx = 0$, it asserts t is fake and informs t of authentication failure. Otherwise, R refers to $KT[idx]$ in

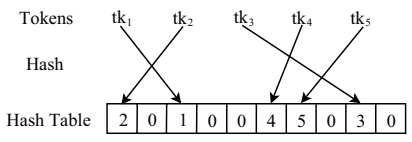

Fig. 2: An illustration of the hash table used by DTAP.

1063-6927/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDCS.2015.94

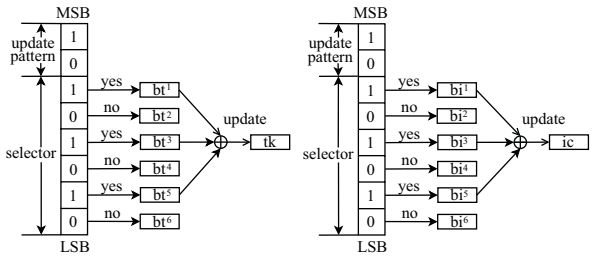

Fig. 3: *Left plot*: Generation of a new token. *Right plot*: Generation of a new indicator.

the key table to retrieve the token, and compares it with the received token tk. Only if the two tokens are the same, will t pass the authentication; otherwise, it fails the authentication. In case that t is successfully authenticated, R will transmit a newly generated token to authenticate itself. The same new token will also be generated by t. The generation of dynamic tokens requires the reader (more exactly, the central server) and the tag to update their shared keys synchronously.

C. Updating Phase

We develop a new technique to update keys and generate random tokens from the base tokens dynamically. Tag t relies on its indicator ic to update its keys. A b-bit indicator consists of two parts: The low-order $(b-2)$ bits form a selector, indicating which base tokens/base indicators should be used to derive the next token/indicator, while the high-order two bits encode the update pattern. When the updating phase begins, t calculates a new token from the base tokens based on the selector. Each of the low-order m bits ($m \leq b-2$) in the selector encodes a choice of the corresponding base token: '0' means *not selected*, while '1' means *selected*. For all selected base tokens, they are XORed to yield a new token. The left plot in Fig. 3 gives an example of token update, where bt^1, bt^3 and bt^5 among the six base tokens happen to be selected to derive the new token. Similarly, t derives a new indicator from the base indicators, as shown in the right plot of Fig. 3. At the reader's side, the same new token and new indicator can be generated because it shares the same keys with the tag. In addition, the reader (central server) also needs to update the hash table. First, the reader sets $HT[h(tk)]$ (the old token) to 0, and after generating the new token, it sets $HT[h(tk)]$ to idx.

After the token and the indicator have been updated, the reader and the tag need to further update the selected base tokens and base indicators. The update process for any selected base token/indicator includes two steps: A one-bit left circular shift is performed first, followed by bit flip based on the particular 2-bit update pattern specified by the indicator: (1) Pattern $(00)_2$: no flip is performed; (2) Pattern $(01)_2$: flip the jth bit if $j \equiv 0 \pmod 3$; (3) pattern $(10)_2$: flip the jth bit if $j \equiv 1 \pmod 3$; (4) Pattern $(11)_2$: flip the jth bit if $j \equiv 2 \pmod 3$. This scheme ensures that any two bits have a chance to not be flipped together if the size of the base tokens/indicators is set properly [1], thereby reducing the mutual dependence of the two bits.

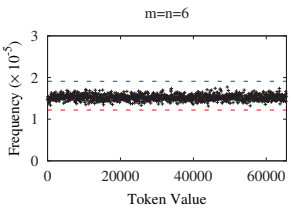

Fig. 4: Frequency test for randomness, where $a = b = 16$, and $m = n = 6$.

D. Discussion

1) Communication overhead: For each authentication, the tag only needs to transmit one token. The reader needs to send an authentication request and one token. The communication overheads for both the tag and the reader are $O(1)$.

2) Online computation overhead: For each authentication, the tag needs to update its keys twice to generate two tokens, and preform one comparison to authenticate the reader. All operations performed by the tag, including bit-wise XOR, bit flip and one-bit circular shift, are very simple. Besides these computations, the reader needs to calculate two hash values: one for the token received from the tag, and the other for the token prepared for next authentication.

E. Randomness

The EPC C1G2 standard [2] requires that for a 16-bit pseudorandom generator the probability of any 16-bit number $RN16$ with value v being drawn from the generator shall be bounded by $\frac{0.8}{2^{16}} < P(RN16 = v) < \frac{1.25}{2^{16}}$. To evaluate the randomness of tokens generated by DTAP, we produce $2^a \times 100$ tokens and calculate the frequency of each distinct token. Limited by the computing power of our computers, we just consider a simple case of $a = 16$ (Note that a longer token can be obtained by concatenating several short tokens). The size b of base indicators is also set to 16 bits, and $m = n = 6$. From Fig. 4, we can see that the tokens generated by DTAP meet the randomness requirement, where the dotted lines represent the required frequency upper and lower bounds.

III. Conclusion

In this paper, we propose an efficient anonymous authentication protocol for RFID systems. To meet the hardware constraint of low-cost tags, we follow the asymmetry principle in our design. Using random sampling and updating techniques, our protocol generates random tokens for anonymous authentication. Moreover, our protocol reduces the communication overhead and online computation overhead to $O(1)$ per authentication for both the tags and the readers, which compares favorably with the prior art.

References

[1] M. Chen, S. Chen, and Q. Xiao, "Pandaka: A Lightweight Cipher for RFID Systems," *Proc. of IEEE INFOCOM*, April-May 2014.

[2] *EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communications at 860MHz-960MHz, EPCglobal*, Available at http://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_1_2_0-standard-20080511.pdf, Version 1.2.0.